

REMARKS

Claims 1-5, 12-13, and 15-27 are now pending. Applicant has cancelled claims 6-11 and 14 and amended claims 1-5, 12, and 16-27.

The Examiner has rejected claims 16-27 under 35 U.S.C. § 101 as being directed to non-statutory subject matter. Although applicant disagrees, applicant has amended these claims to address the Examiner's concerns.

The Examiner has rejected claims 16-27 under 35 U.S.C. § 112, second paragraph, as omitting essential elements. Applicant respectfully disagrees. The preamble of the claims recite that the computer-readable storage medium contains instructions. The bodies of the claims recite the method that is performed by the instructions, which are contained on the computer-readable storage medium. The "element" of the claims is the instructions, which are limited by the bodies.

The Examiner has rejected the claims as follows:

Claims	Statute	References
1-10 and 16-18	35 U.S.C. § 102(e)	Kobayashi
11 and 20-26	35 U.S.C. § 103(a)	Kobayashi and Rodrigues
12-15	35 U.S.C. § 103(a)	Kobayashi, Rodrigues, and MTS
19	35 U.S.C. § 103(a)	Kobayashi
27	35 U.S.C. § 103(a)	Kobayashi and Gilgen

Although applicant disagrees, applicant has amended the claims to clarify the subject matter of the invention.

Applicant would like to thank the Examiner for his consideration of and comments on the arguments provided by applicant on February 3, 2006. This response reiterates those arguments and provides additional arguments.

Kobayashi describes a technique for creating an application using a visual programming tool. The technique creates a proxy component, also referred to as a "bean" or "proxy bean," for each method of a component class code. Thus, each bean only has one method associated with it. Kobayashi describes the creating of beans as follows. "The compiler 306 also creates a method bean from extracted information for each method in the class, for example method beans 310-316." (Kobayashi, 8:56-58.) "[T]he bean compiler 300 [sic, 306] extracts the actual parameter names. . . . Extracting these parameter names allows these parameters to be converted to properties of the method bean created from the original method. . . ." (Kobayashi, 9:12-16.)

A programmer can then select which beans are to be included in the application, also referred to a "composite component" and "composite bean," and set the properties of the beans. A property of a bean can either be constant or variable. Figure 11 illustrates the setting of constant values for the properties of a bean. In this example, the label property has the constant value "new bean db," and the font property has the constant value "arial." (Kobayashi, Fig. 11, 1155.) Figure 17 illustrates the setting of variable values for the properties of a bean. Kobayashi refers to the setting of the variable values as a "wiring" the beans. In Figure 17, the foreground property of the button bean is the source of the variable value that is to be connected to the background property of the chart bean as the target of the variable value. During run time, when a bean is executed, its underlying method of the underlying object is invoked with the values, constant or variable, of its actual parameters.

To run an application, the beans of the application "instantiate" constructor and method objects to invoke the appropriate constructors and methods for the target class in the implementation code (i.e., the underlying objects). (Kobayashi, 9:40-49.) Figure 10 illustrates the process of instantiating objects and invoking of methods of those objects. Method bean 1012 includes an invocation of a constructor 1014, an invocation of a method 1016, and a storing of the result 1017 of the invocation of the method. When the method bean 1012 of the application is invoked, it first invokes the constructor to construct or

instantiate the underlying object, if not already instantiated. The underlying object is represented as being stored in Java class implementation 1048. The method bean 1012 then invokes 1016 the method which causes the corresponding method 1056 of the instantiated underlying object to be invoked.

Figure 20 of Kobayashi illustrates the testing of an application. Steps 2006-2014 illustrate that to test an application the programmer first runs 2006 the application. The beans of the application control the instantiating of the underlying objects and invoking 2008 of the methods of the underlying objects. If the programmer decides 2012 that the application did not work correctly, the programmer edits 2014 the application and repeats the process. By repeating the process, the underlying objects are again instantiated and the methods again invoked in accordance with the edited application.

Applicant's claims are directed to a technique for testing a software object. For example, claim 16 recites instantiating an object and then exercising the instantiated object. The instantiated object is exercised by displaying methods of the object to the user and allowing the user to select a method and to provide parameters for the method. The selected method of the instantiated object is then invoked passing the parameters.

It is unclear to applicant whether the Examiner believes Kobayashi's beans or their underlying objects correspond to applicant's instantiated object.

Applicant wishes to point out, however, that Kobayashi's beans are created by the bean compiler to have only one method. Thus, Kobayashi would have no reason to display "methods" of the bean as would be required to meet one of the limitations of claim 16.

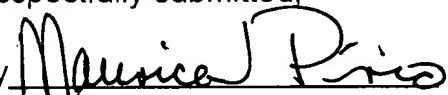
Applicant also wishes to further point out that Kobayashi neither teaches nor suggests the testing of the underlying objects. Rather, any testing suggested by Kobayashi is to test the application created by the visual programming tool. Kobayashi assumes that the Java class implementations function properly.

Claims 1-5 now recite "instantiating an object" that has methods, "retrieving from an input dialog a selected method," and "obtaining an actual parameter" for the selected method. Claims 12-13 and 15 recite "displaying to a user a list of methods of the object." Claims 22-27 recite "each object having methods" and "invoking the method of the entry of the instantiated object." Thus, all these claims make it clear that an instantiated object has multiple methods and that user input or entries indicate what method to test with what actual parameter. Moreover, the claims recite a novel combination of steps that is neither taught nor suggested in the cited references.

Based on the above amendments and remarks, applicant respectfully requests reconsideration of this application and its early allowance. If the Examiner has any questions or believes a telephone conference would expedite prosecution of this application, the Examiner is encouraged to call the undersigned at (206) 359-8548.

Dated: March 13, 2006

Respectfully submitted,

By 
Maurice J. Pirio

Registration No.: 33,273
PERKINS COIE LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000
(206) 359-7198 (Fax)
Attorneys for Applicant